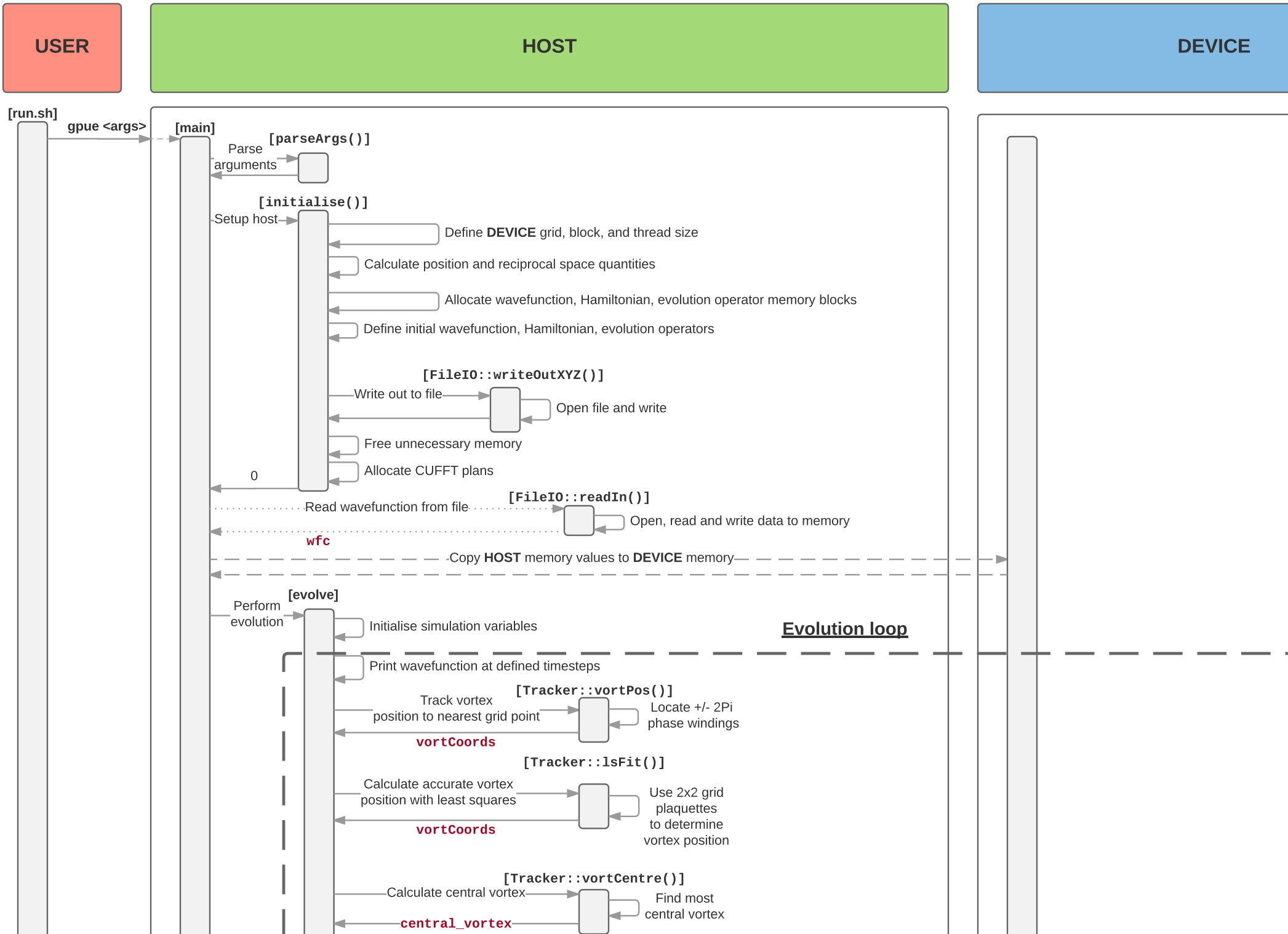
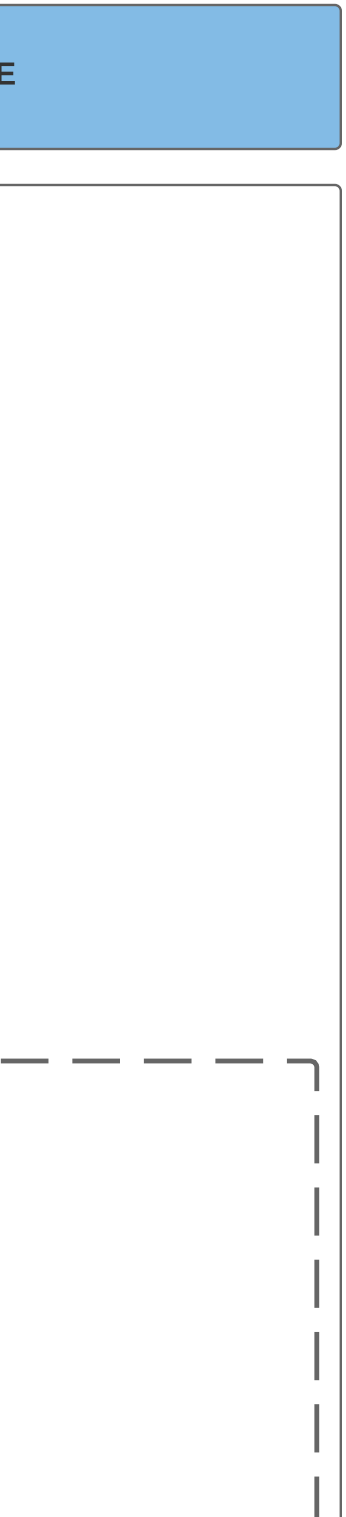
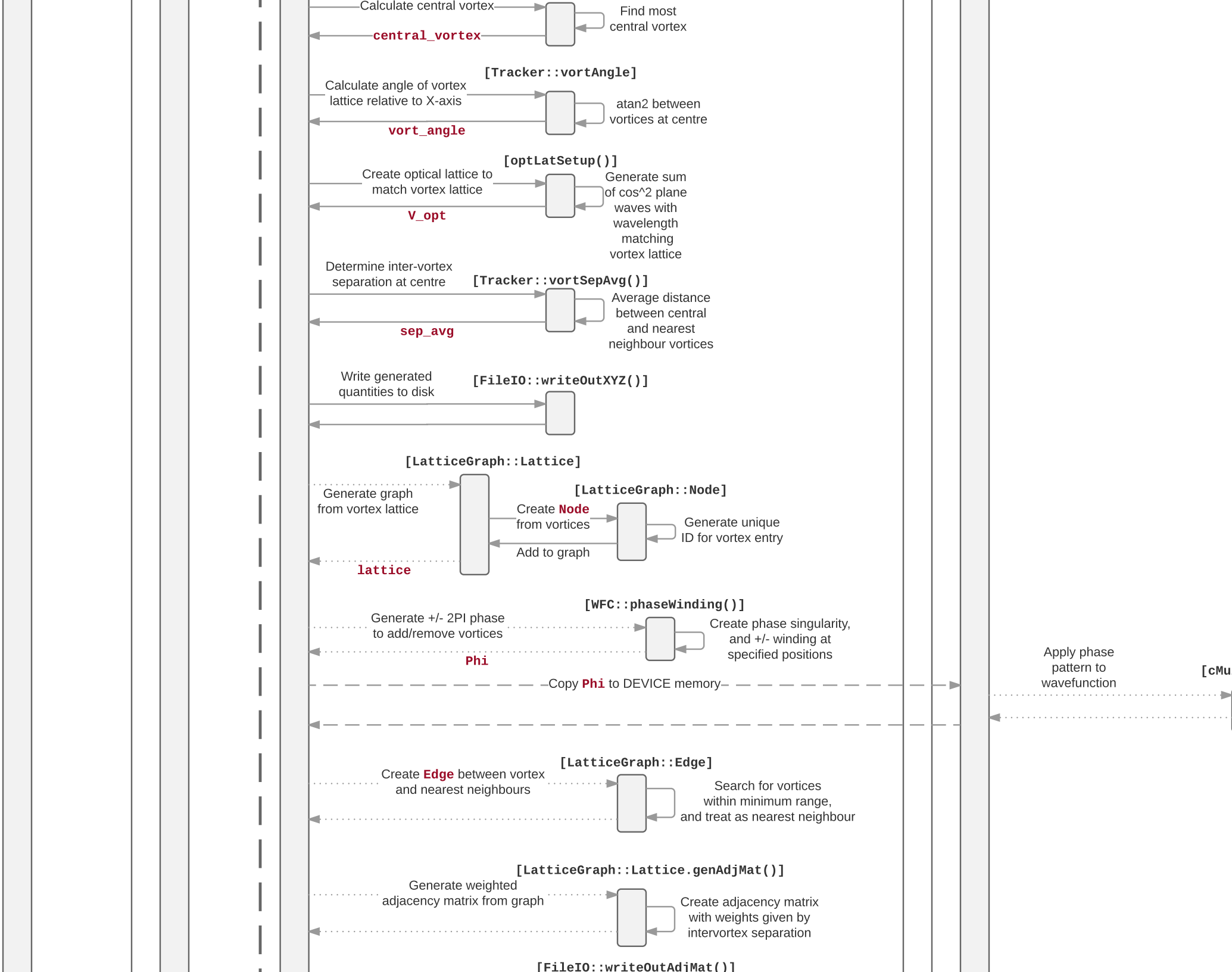
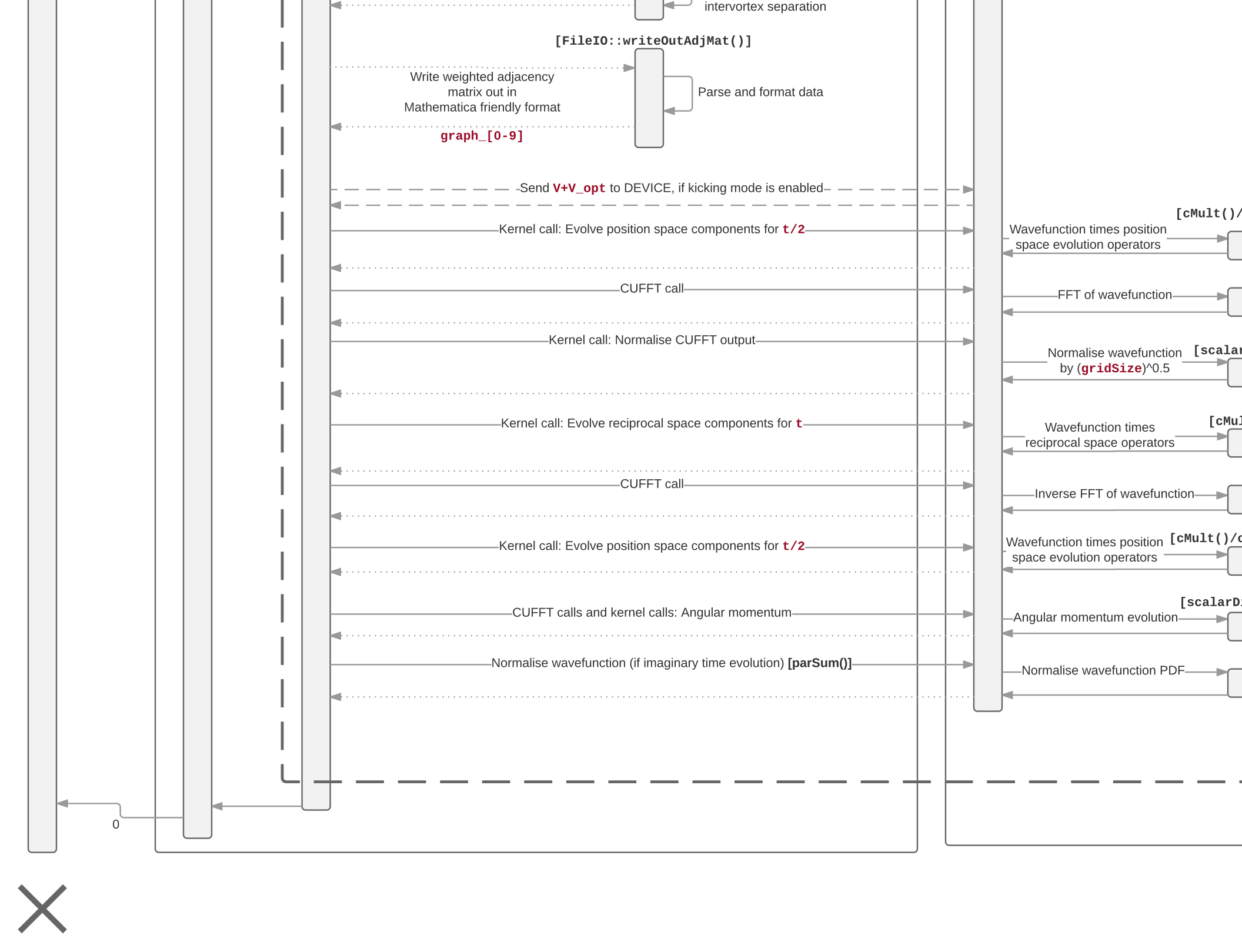
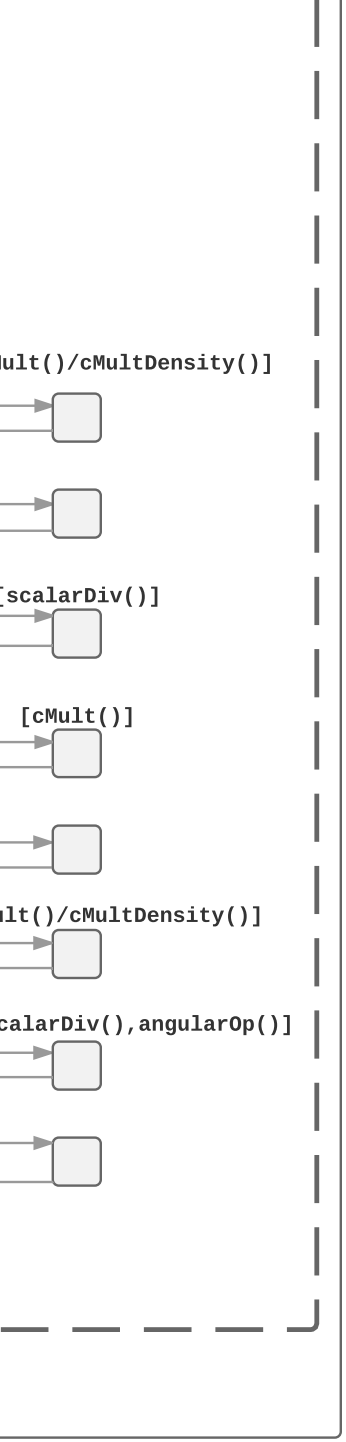

USER
HOST
DEVICE
[run.sh]
gpue <args>
[main]
Parse arguments
[parseArgs()]
[initialise()]
Setup host
Define DEVICE grid, block, and thread size
Calculate position and reciprocal space quantities
Allocate wavefunction, Hamiltonian, evolution operator memory blocks
Define initial wavefunction, Hamiltonian, evolution operators
[FileIO::writeOutXYZ()]
Write out to file
Open file and write
Free unnecessary memory
Allocate CUFFT plans
0
[FileIO::readIn()]
Read wavefunction from file
Open, read and write data to memory
wfc
Copy HOST memory values to DEVICE memory
[evolve]
Perform evolution
Initialise simulation variables
Evolution loop
Print wavefunction at defined timesteps
[Tracker::vortPos()]
Track vortex position to nearest grid point
Locate +/- 2Pi phase windings
vortCoords
[Tracker::lsFit()]
Calculate accurate vortex position with least squares
Use 2x2 grid plaquettes to determine vortex position
vortCoords
[Tracker::vortCentre()]
Calculate central vortex
Find most central vortex
central_vortex




Calculate central vortex
Find most central vortex
central_vortex
[Tracker::vortAngle]
Calculate angle of vortex lattice relative to X-axis
atan2 between vortices at centre
vort_angle
[optLatSetup()]
Create optical lattice to match vortex lattice
Generate sum of cos^2 plane waves with wavelength matching vortex lattice
V_opt
Determine inter-vortex separation at centre
[Tracker::vortSepAvg()]
Average distance between central and nearest neighbour vortices
sep_avg
Write generated quantities to disk
[FileIO::writeOutXYZ()]
[LatticeGraph::Lattice]
Generate graph from vortex lattice
[LatticeGraph::Node]
Create Node from vortices
Generate unique ID for vortex entry
Add to graph
lattice
[WFC::phaseWinding()]
Generate +/- 2PI phase to add/remove vortices
Create phase singularity, and +/- winding at specified positions
Phi
Copy Phi to DEVICE memory
Apply phase pattern to wavefunction
[cMu
[LatticeGraph::Edge]
Create Edge between vortex and nearest neighbours
Search for vortices within minimum range, and treat as nearest neighbour
[LatticeGraph::Lattice.genAdjMat()]
Generate weighted adjacency matrix from graph
Create adjacency matrix with weights given by intervortex separation
[FileIO::writeOutAdjMat()]


[cMultPhi()]


intervortex separation
[FileIO::writeOutAdjMat()]
Write weighted adjacency matrix out in Mathematica friendly format
Parse and format data
graph_[0-9]
Send V+V_opt to DEVICE, if kicking mode is enabled
Kernel call: Evolve position space components for t/2
Wavefunction times position space evolution operators
CUFFT call
FFT of wavefunction
Kernel call: Normalise CUFFT output
Normalise wavefunction by (gridSize)^0.5
Kernel call: Evolve reciprocal space components for t
Wavefunction times reciprocal space operators
CUFFT call
Inverse FFT of wavefunction
Kernel call: Evolve position space components for t/2
Wavefunction times position space evolution operators
CUFFT calls and kernel calls: Angular momentum
Angular momentum evolution
Normalise wavefunction (if imaginary time evolution) [parSum()]
Normalise wavefunction PDF
0


Mult()/cMultDensity()]
[scalarDiv()]
[cMult()]
lt()/cMultDensity()]
calarDiv(),angularOp()]